Computer Networks and ISDN Systems 29 (1997) 917–926

COMPUTER NETWORKS and ISDN SYSTEMS

Promondia: a Java-based framework for real-time group communication in the Web

Ulrich Gall [a,b,*], Franz J. Hauck [a,1]

[a] IMMD IV, Department of Computer Science, University of Erlangen-Nürnberg, Erlangen-Nürnberg, Germany
[b] 3SOFT GmbH, Erlangen, Germany

Abstract

The World Wide Web has evolved from a distributed hypertext system to a platform-independent graphical user interface that integrates many network services. So far, its technology has restricted it mainly to applications for information retrieval.

As networks become ubiquitious and more and more users have a permanent connection, there is an increasing demand for other network services, such as real-time data feeds, group communication, and teleconferencing. So far, these services have been provided by various proprietary software systems, which were hard to set up and use, and thus not very successful.

Integrating real-time group communication services into the World Wide Web is a natural way to make them more accessible and will take the Web a step further on its way to becoming the universal network application.

In this paper, we describe functionalities required for these services and present an implementation based on Sun Microsystem's **Java** [2] programming language. We focus on the high-level functionalities and abstractions, but also describe an object-oriented programming model for group communication systems. © 1997 Published by Elsevier Science B.V.

Keywords: Integration of real-time channels; Innovative applications; Programmatic and specialized user interfaces

---

1. Overview

Most people agree that communicating is the most important activity in today's information society. We spend a very significant portion of our time — work and leisure — communicating in groups with other people, either directly (face to face), or remotely (via telephone, email, or videoconferences). Therefore, finding more efficient means for communication between people is a very important area of research. We believe that the most significant problem that keeps people from communicating with one another via teleconferencing software systems is their limited accessibility.

The World Wide Web has become the most convenient way to access information in the Internet. The main reason for this is that WWW browsers integrate different network services into a common, easily accessible, platform-independent user interface.

Developing group communication services that support conferencing from within the WWW is

---

[*] Corresponding author. E-mail: uhgall@informatik.uni-erlangen.de

[1] E-mail: hauck@informatik.uni-erlangen.de

[2] http://www.javasoft.com

0169-7552/97/$17.00 © 1997 Published by Elsevier Science B.V. All rights reserved.
PII S0169-7552(97)00007-X

an excellent way to improve accessibility of such services. This should be done in a way that does not require specific proprietary features of WWW browsers; it should be built on general mechanisms provided by all browsers.

So far, the WWW does not support most of the features required for these kinds of applications. For example, there is no mechanism for server-initiated activity. HTTP is not intended for real-time data feeds — it is state- and connectionless. Using CGI scripts and forms, the user interacts with the Web server rather than the browser. Active contents, such as Java Applets, solve these problems. They are a very general mechanism supported by most browsers, yet they are powerful enough to develop serious network-aware applications.

In this paper, we will describe some general design issues for group communication systems for the WWW and present an implementation of such a system. The paper is organized as follows. Section 2 provides a short introduction to group communication. In Section 3, we discuss some design issues for the system. We present our proof-of-concept implementation, called *Promondia*, in Section 4 and discuss related work in Section 5. Finally, Section 6 summarizes our results.

2. Group communication

Group communication is the exchange of information between a group of participants in a *session*[3]. Participants may have different roles in a session. Information is exchanged using certain media, chosen according to availability and suitability for the information to be transferred.

Different types of group communication sessions have evolved in the past in order to reach common goals as efficiently as possible. These goals include entertainment, decision making, learning, and plain information exchange about a common interest.

For example, a common pattern is a panel discussion involving a host, several speakers, and an audience (see Fig. 1). Typically, all these participants are in the same room. Host and speakers communicate to the audience and among one another using audio and gestures. The audience provides feedback using facial expressions and gestures. The host may let individuals ask questions. The goal is to communicate information the audience is interested in.

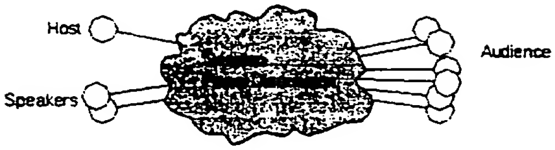


Fig. 1. Session of a panel discussion.

This type of group communication is based mostly on the availability of media, not on their suitability. New technology provides the opportunity to choose more appropriate media to reach a given goal more efficiently. Using this new technology in the example above could mean that the audience, distributed all over the world, can provide detailed feedback without disrupting the speakers. This feedback can be summarized automatically, the host and the speakers can see immediate statistics and react accordingly. Anonymous questions can be submitted to the host, who could choose to forward them or discard them.

Group communication can roughly be divided into two categories, synchronous and asynchronous:

	Synchronous	Asynchronous
Time frame	"Real-time" Seconds	Delayed Minutes to indefinitely
Transfer of information	When available	When requested
Analogy in traditional media	Telephony (bidirectional), life TV (unidirectional)	Voice mail, Bill boards
New technologies	Videoconference, telephony, shared whiteboard	E-mail, threaded discussion systems
Products	***ShowMe***[4] **(SUN Microsystems**[6]**)** ***ProShare***[7] **(Intel Inc.**[8]**)**	*Lotus Notes* **(Lotus**[5]**)** Databases in general

---

[3] In the Internet-Relay-Chat (IRC) [1] this is referred to as a *channel*, in most Web-based Chat systems as a *room*.

[4] http://www.sun.com/products-n-solutions/sw/ShowMe/

[5] http://www.lotus.com

[6] http://www.sun.com

[7] http://www.intel.com/comm-net/proshare/

[8] http://www.intel.com

Asynchronous communication can be viewed as synchronous communication with one virtual participant (like a VCR recording a TV show) that stores information and provides them when requested by another participant. Thus, synchronous communication is the more general concept of the two.

Asynchronous communication can easily be integrated into the WWW using HTML forms or Java front ends and CGI scripts or database servers. Many products and services have been developed for this.

Uni-directional real-time data feeds can also be seen as a special case of group communication with one or more participants, which are simply data sources, and a large number of passive recipients.

In this paper, we focus on real-time, synchronous group communication and on how it can be integrated into the World Wide Web.

3. Design aspects for group communication in the WWW

Most systems for group communication in use today do not clearly separate all of the four components that make up their characteristics: network topology, network services, programming models for and implementations of session types.

There should be a clear distinction between these components. We will discuss the first three in this section and the last one in Section 4.

3.1. Client/server approach

Certainly, it is helpful to make a clear distinction between a server subsystem and clients when designing an architecture for group collaboration in the WWW. The server subsystem provides functionality common to all session types, such as logging, access control, and directory services. For each session, it also contains a session-server object that coordinates the communication between the clients and determines the contents distributed in that session. This session-server object may be generic or session-type specific.

A client runs on the participant's system and provides the front end that presents the session to the participant. Different types of clients reflect the different roles for each session type: when a participant joins, the appropriate client is instantiated.

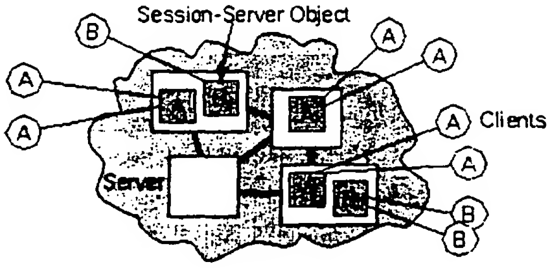


Fig. 2. Servers with different session-server objects.

For sessions with a large number of participants (> 1000), a scalable architecture is required. This can be achieved using a distributed server subsystem. A network of servers share a common name space and host server proxies on behalf of their clients. These proxies are also session-server objects. They may be generic and simply forward information to and from a central main server, or they may contain semantics for the type of session they serve (e.g., they summarize the local results of a voting session).

Since the the session-server objects may be loaded via the net and instantiated dynamically, a security model is required that restricts their access to local resources. Therefore, Java's coarse-grained security model is not suitable. The more fine-grained security model described in [2] may be appropriate.

3.2. Underlying network services

The system should be based on a reliable network service to set up the software components and to exchange session and user management data. However, for the actual information to be communicated, it should be possible to use other network services based on the requirements of the particular type of communication. In this case, the system network is used to exchange the references required to use the other network services.

Network services that could be used include TCP, UDP, IP Multicast, and more specialized protocols such as RTP [3] or TIBCO Inc.'s Rendezvous information bus [4]. A very early version of our system [5] used IRC [1] as the messaging backbone. CORBA Events services [6] could be used for strongly typed, language-independent event multicasting.

3.3. Programming models for implementing session types

Many programming models for real-time collaboration have been suggested. The simplest and most basic mechanism is **message passing** between the objects associated with the participants. A convenient API can offer asynchronous multicast for remote method invocation on all or a subset of participants [7]. This model is very suitable and intuitive to use for volatile communication-oriented tasks, such as chatting. It can be refined by distinguishing certain characteristics for the messages (e.g., unreliable delivery or unspecified delivery order).

To identify different types of messages, it has proven practical to allow for more than one named communication channel per session [8]. Clients interested in a particular type of message can subscribe to its respective channel. That way, the different clients reflecting the different roles only receive the information they need.

To allow for a *reply* to the messages sent, delayed return values for asynchronous method invocation can be implemented by creating a handle to represent the reply expected in the future. This reference can then be used to wait for the reply or to forward it somewhere. An extension of Java providing this is *E*, described in [9].

Other, more sophisticated models can be built on top of this simple infrastructure.

Model-View-Controller-based (MVC) approaches with a replicated model, one view and one controller per user are very suitable for collaboratively creating or modifying data. However, they tend to be inefficient if all changes to the model are immediately sent to all views. However, Graham et al. report on significant performance advantages using optimized network communication [10]. If a more sophisticated strategy is used to transmit changes only when needed, application specific semantics have to be considered. This can be quite difficult to implement.

Gutfreund et al. suggest a Linda-like simplified **tuple space** model for coordinating distributed applications [11,12]. This is a very lightweight approach to the MVC pattern that provides a generic server and requires all semantics in the client objects. This can also be used for simple message passing, since clients may register for changes on tuples. **Lotus**[9] has a similar approach, using "things" as state-keeping objects that also provide notification services [13].

For synchronisation and floor control, **token-based mechanisms** can be used to ensure data integrity [14]. This is an easy way to use collaboration-unaware programs collaboratively, but severely restricts interaction between participants.

Finally, real-time multimedia streaming requires sophisticated network protocols with different quality of service characteristics. It should be possible to integrate these services and have them controlled by the application.

All these different approaches are very suitable for certain tasks, whereas they are inconvenient or inefficient for others. We believe that these models should complement each other rather than compete. A framework for group communication should not be bound to a particular model, but provide different models for the developer to choose from.

To allow for software reusability, it should be possible to combine different session types to form a new type. This can be done recursively to create a hierarchy of components and to dynamically add new ways of communication to a session when required. When a client joins a session containing a subsession, it automatically joins the subsessions available in that session, too.

3.4. Integration into the Web

For integrating group communication into the Web, a session should run within a Web page using standard Web browsers. That way, they can be combined with HTML data containing general information about the session or its topic. Also, sessions can be embedded into WWW pages about a particular subject to let people, interested in this subject, communicate with each other and create a virtual community.

Standard WWW mechanisms can be used for directory services and bookmarking. Sessions can be embedded into MIME email messages so that they are automatically started when the recipient reads

---

[9] http://www.research.lotus.com

the email message. This is a convenient way to send invitations for sessions. Within a session, Web pages can be shown to all participants to point to information relevant for the current topic.

4. Promondia: a proof-of-concept implementation

We have implemented a system, called *Promondia*, providing group communication functionality for the WWW. An early version of our system is known under the name *COMO*.

Promondia consists of a server program and session starters implemented as Java applets, which are embedded in HTML documents. These applets are called session-management applets.

When the user views an HTML document that contains a reference to a session-management applet, the browser loads this applet via HTTP and starts it. The applet does not necessarily have to be located on the same WWW server as the HTML page containing it. However, access via certain WWW servers can be restricted.

When the applet is started, it connects to the *Promondia* server and requests to join the specified session. If that session does not exist, a new session is created on the server by instantiating the specified session-server object. When a session is joined, the session-starter applet instantiates the session client reflecting the participant's role in the session.

4.1. Session server

The Promondia server is a stand-alone Java application. It is entirely written in Java and does not require any native code. Thus, it runs on any platform for which a Java interpreter is available.

The server includes an HTTP server. This is very convenient for the following reason: Current Web browsers allow applets to connect only to the host they were loaded from. Otherwise, it would be possible to bypass a firewall with a malicious applet. Thus, the Java applets must be loaded from the system the Promondia server is running on. This can be achieved by using the built-in HTTP server. As an alternative, a separate standard HTTP server can be set up on the same machine.

A group-management module organizes and coordinates the group-communication channels. It also provides information about active sessions. It supports access control on a general, a per-session type, and a per-session basis. Configuration can be done remotely using special Java applets.

The multithreaded architecture makes up for performance weaknesses that are due to the fact that the Java byte code has to be interpreted. Without just-in-time compilation, a 133 Mhz Pentium PC can handle about 100 simultaneous connections. A Sun UltraSparc workstation can handle several hundred connections at the same time. Using Kaffe [15], a freely available runtime system for Java, 350 simultaneous connections have been handled easily on a Pentium 120 running Linux.

4.2. Session management applets

Two session management applets have been implemented to embed sessions into an HTML document:

- A *PageStarter* simply joins a specified session and displays its GUI within a Web page. Visual characteristics can be specified in the HTML document.
- A *FloatStarter* can include advertisements or a help menu and lets a user detach the applet from the Web page by opening a separate window. Thus, the session can be run independently from the Web page.

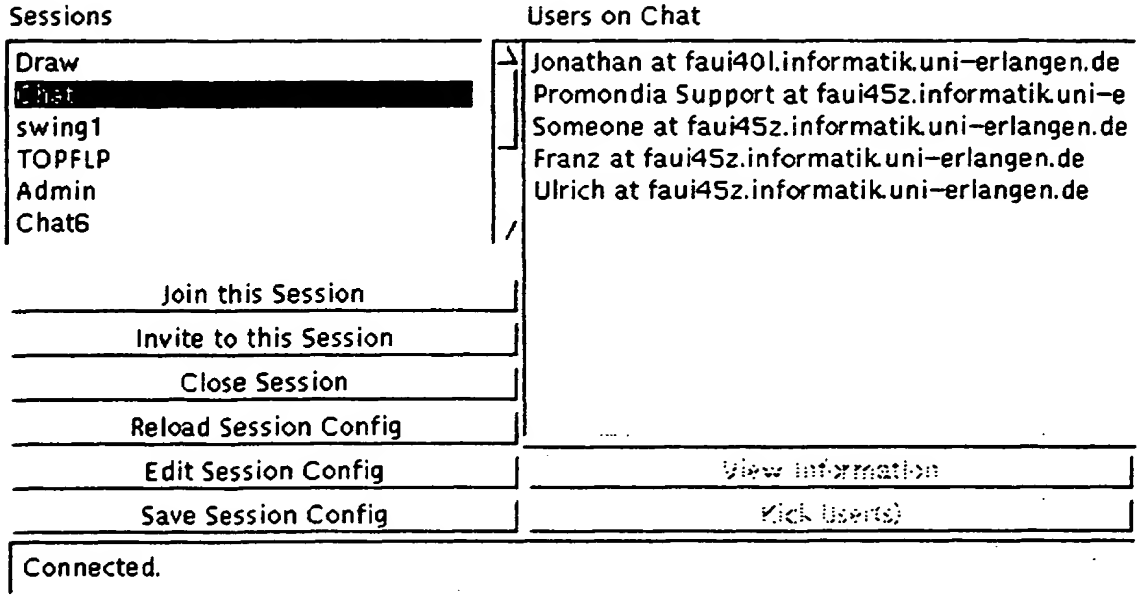
Other session management modules have been implemented as sessions. That way, they can use the communication mechanisms provided by the system to distribute information about available users and session.

4.3. Implemented session types

A few session managers and communication methods have been implemented to demonstrate the system.

4.3.1. Manager Session type

A *Manager Session* provides information about which sessions are running and which users are available. This is an example of how a session can be used for multicast distribution of information.



Draw
Chat
swing1
TOPFLP
Admin
Chat6

Jonathan at faui40l.informatik.uni-erlangen.de
Promondia Support at faui45z.informatik.uni-e
Someone at faui45z.informatik.uni-erlangen.de
Franz at faui45z.informatik.uni-erlangen.de
Ulrich at faui45z.informatik.uni-erlangen.de

Connected.

Fig. 3. The *Manager-Session* client for administrators.

It provides two roles: *user* and *administrator*. A user can select sessions and join them, which will download the client code for the session and spawn a new window. Invitations can be sent via email; the recipient can join the session simply by reading the email with a Java-enabled browser. The administrator can change the properties and configuration of the sessions and expel users from a session. Fig. 3 shows a screenshot of a *Manager-Session* client.

4.3.2. Text-based Chat

The *Chat* system offers a variety of chat modules: a simple one and several more sophisticated ones, which allow for private and anonymous messages, notification at certain events, and audio messages. Users can direct other participants to Web pages by typing a URL. Fig. 4 shows a *Chat* client with an ongoing session. A *Moderator* client lets the administrator control what is being said in the session by assigning different rights (free speech, moderated, muted) on how to take part in the conversation.

Promondia Chat sessions can be connected to IRC channels. The information transferred is mapped both ways. If the session is moderated, the IRC channel will also be moderated.

4.3.3. Shared Whiteboard

The *Shared Whiteboard* is a multi-user vector-oriented drawing program. Lines, polygons, rectangles, ovals, text, and imported images can be added to a shared canvas (see Fig. 5). This kind of functionality can be used as a subsession of a conferencing system to illustrate ideas or discuss designs.

4.3.4. Voting and surveys

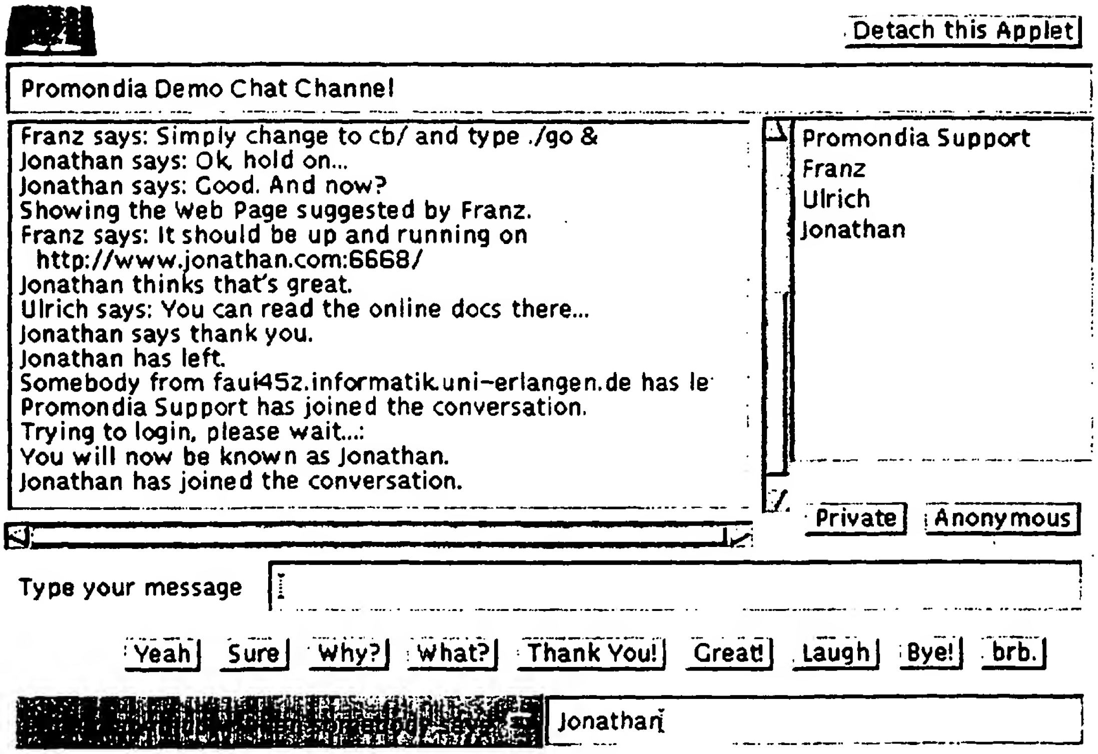
A *VoteModerator* applet lets an administrator configure a question and a set of suggested answers. The users can choose one of the answers to cast their votes using a *Voter* client. The results can be transmitted automatically or after approval by the moderator. In Fig. 6, a *Voter* is shown. The already known votes are presented as colored bars behind the suggested answers.

4.3.5. Games

A **simple board game**[10] for 2 to 10 players has also been implemented to illustrate that multi-player games can be seen as a form of group communication with very formal interaction rules.

---

[10] http://www4.informatik.uni-erlangen.de/promondia/products/promondia/www/doc/NNN.html



Promondia Demo Chat Channel

Franz says: Simply change to cb/ and type ./go &
Jonathan says: Ok, hold on...
Jonathan says: Good. And now?
Showing the Web Page suggested by Franz.
Franz says: It should be up and running on
http://www.jonathan.com:6668/
Jonathan thinks that's great.
Ulrich says: You can read the online docs there...
Jonathan says thank you.
Jonathan has left.
Somebody from faui45z.informatik.uni-erlangen.de has le
Promondia Support has joined the conversation.
Trying to login, please wait...:
You will now be known as Jonathan.
Jonathan has joined the conversation.

Promondia Support
Franz
Ulrich
Jonathan

Type your message

Jonathan

Fig. 4. A *Chat* client.

5. Related work

Most work on Web-based collaboration focuses on asynchronous database- and transaction-oriented cooperation. An example is the *BSCW* system [16,17] developed as part of the *CoopWWW* project [18]. Current implementations of synchronous communication for the Web are mostly limited to chats or other predefined functionality.

GroubWeb [19] is a collaborative Web Browser that is built on top of *GroupKit* [7]. This approach can even make Web browsing a fully collaborative activity; however it does not improve accessibility of group communication, since proprietary software has to be installed. The same applies to RAVE [20], which uses browser plug-ins for presenting real-time data feeds. CoWeb [21] solves these problems by using Java, but it is not programmable.

GroCo is a simple, but flexible system based on message passing and implemented for the alpha release of Java [22,23]. The data objects sent in the messages are restricted to certain types.

Mushroom is a very promising and ambitious project that focuses on an intuitive user interface. A *room* metaphor is used to represent a session. Persistent data can be stored in a room for asynchronous cooperation. Users may enter a room and meet for synchronous cooperation [24,25]. A Java-based implementation is under way, but not available yet.

JavaSoft is working on a collaboration API [14] that is based on ITU recommendation T.122 [8]. This API does not yet cover higher-level functionality for session management and configuration.

6. Conclusion and future work

We have discussed some of the issues involved in designing an architecture for a group communication infrastructure for the World Wide Web. We believe

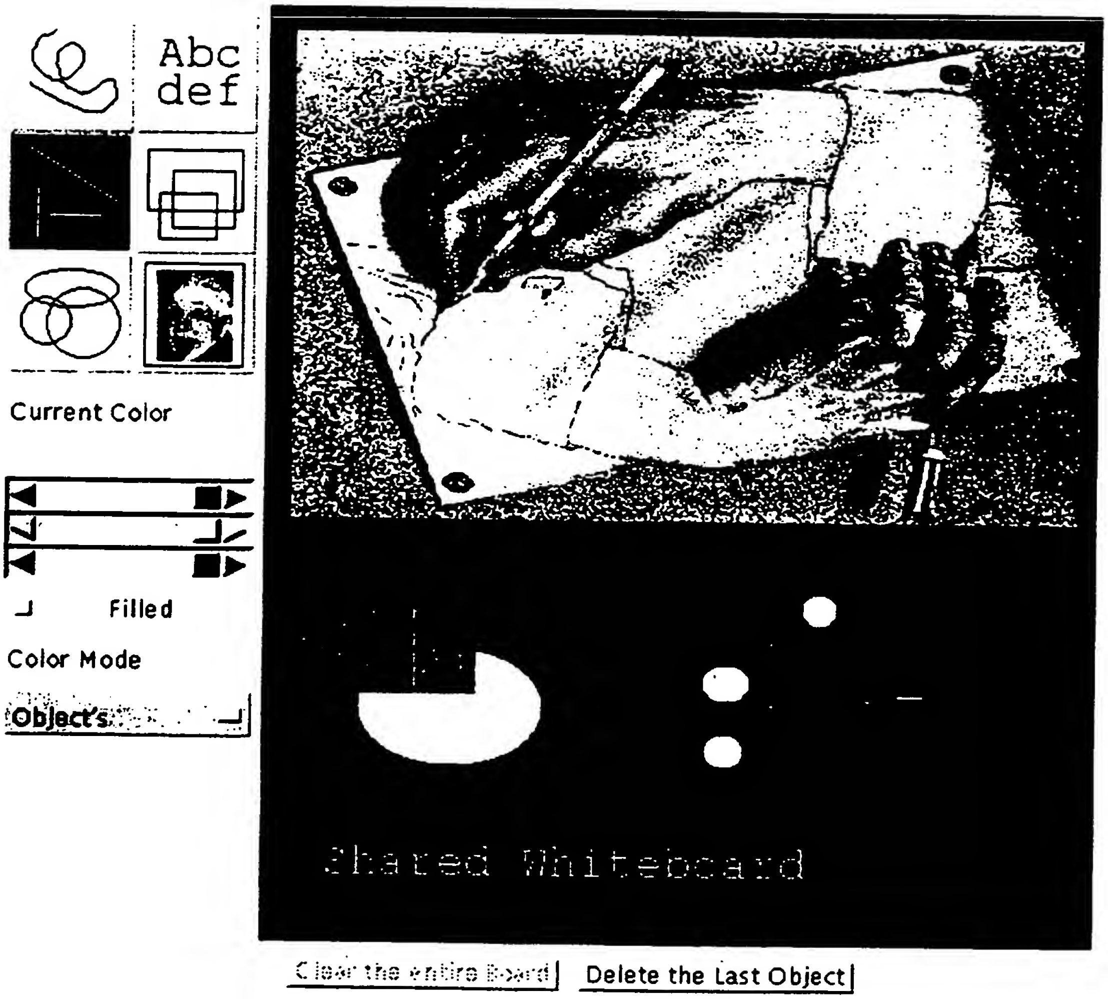
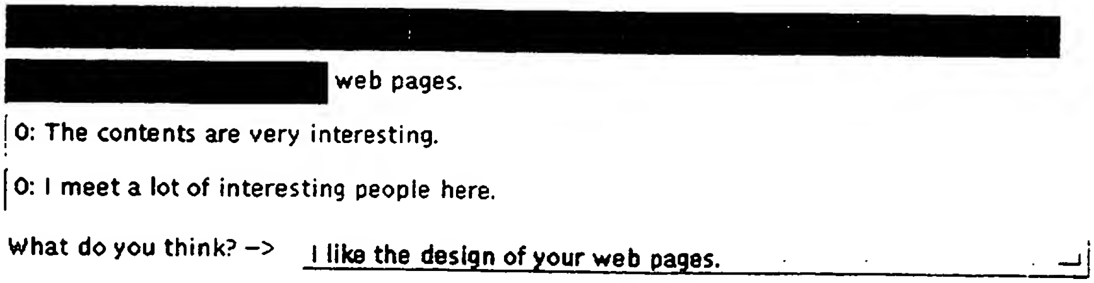


Fig. 5. The *Shared Whiteboard* client with a loaded image and some graphics.



[redacted]

[redacted] web pages.

0: The contents are very interesting.

0: I meet a lot of interesting people here.

What do you think? –> I like the design of your web pages.

Fig. 6. A *Voter* client.

that building such an infrastructure can help people to communicate more efficiently, since it would make these application more accessible.

Promondia, our award-winning [11] implementation of such a system is freely available for academic and non-profit organizations at http://www4.informatik.uni-erlangen.de/promondia/. It is used to provide chat services on many public Web sites [12] in the Internet. It is unique in that it is the only flexible, high level infrastructure currently available for building synchronous group-communication applications for the Web. However, it lacks many of the features necessary for a globally usable system.

There is still a lot of work to be done. In particular, we are currently working on a scalable reliable multicast backbone. The programming models that have been suggested in the past can be combined to let the developer choose the model that is most suitable for the task. However, we feel that it would be helpful to integrate these models in a way that allows for component reusability. For that, we are currently investigating the usability of a model with a hierarchy of light weight communication channels and with a very weak distinction between channels and sessions.

References

[1] J. Oikarinen and D. Reed, Internet Relay Chat Protocol, Request for Comments #1459, May 1993, http://www.internic.net/rfc/rfc1459.txt

[2] T. Riechmann, Security in large distributed, object-oriented systems, Techn. Report TR-14-96-02, IMMD IV, Univ. of Erlangen-Nürnberg, June 1996, http://www4.informatik.uni-erlangen.de/TR/ps/TR-14-96-02.ps.Z

[3] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, RTP: a transport protocol for real-time applications, Request for Comments #1889, Jan. 1996, http://www.internic.net/rfc/rfc1889.txt

[4] TIBCO Inc., White Paper [about the Rendevouz Information Bus], 1996, http://www.rv.tibco.com/rv-whitepaper20.html

[5] J. Begole, Review of Como, 1996, http://www.cs.vt.edu/~hci/virtSchool/ComoReview.HTM

[6] R. Orfali, D. Harkey and J. Edwards, *The Essential Distributed Objects Survivial Guide*, Wiley, New York, NY, 1996

[7] M. Roseman and S. Greenberg, Building real time groupware with GroupKit, a groupware toolkit, in: *ACM Trans. on Computer Human Interaction*, ACM Press, March 1996, http://www.cpsc.ucalgary.ca/projects/grouplab/

[8] International Telecommunication Union Recommendation T.122, http://www.itu.int/ITU-Databases/Macbeth/

[9] Electric Communities, *The E programmer's manual*, 1996, http://www.communities.com

[10] T.C.N. Graham, T. Urnes and R. Nejabi, Efficient.distributed implementation of semi-replicated synchronous groupware, in: *Proc. of the ACM Symp. on User Interf. Softw. and Techn.*, ACM Press, Nov. 1996, http://www.cs.yorku.ca/~graham/uist96.html

[11] WWWinda Home Page, http://info.gte.com/ftp/circus/Orchestrator/documentation/WWWinda.html

[12] D. Gelernter, Multiple tuple spaces in Linda, in: *Proc. of PARLE* '89, Springer, Berlin, 1989, pp. 20–27.

[13] M. Day, J. Patterson, J. Kucan, W. Meng Chee and D. Mitchell, Notification Service Transfer Protocol (NSTP), Version 1.0, Techn. Report 96-08, Lotus Inc., 1996, http://www.lotus.com/research/21ae.htm

[14] R. Burridge, Java Shared Data API, 1996, http://java.sun.com/people/richb/jsda/

[15] Kaffe Home Page, 1996, http://www.tjwassoc.demon.co.uk/kaffe/kaffe.htm

[16] R. Bentley, T. Horstmann, K. Sikkel and J. Trevor, Supporting collaborative information sharing with the World Wide Web: the BSCW shared workspace system, in: *Proc. of the 4th WWW Conf.* (Boston, MA, 1995), http://orgwis.gmd.de/~bscw/papers/boston-95/BOSTON.html

[17] W. Appelt and U. Busbach, The BSCW system: a WWW based application to support cooperation of distributed groups, 1996, http://orgwis.gmd.de/~busbach/wetice.ps

[18] W. Appelt, CoopWWW: interoperable tools for cooperation support using the World Wide Web, in: *Proc. of the ERCIM Workshop on CSCW and the Web* (St. Augustin, Germany, Feb. 1996), http://orgwis.gmd.de/projects/W4G/proceedings/coopwww.html

[19] S. Greenberg and M. Roseman, GroupWeb: A groupware Web browser, in: *Video Proceedings of ACM CSCW'96 Conference on Computer Supported Cooperative Work* (Boston, USA, ACM Press, Videotape., 1996). A two page

---

[11] Promondia (formerly called COMO) has won the following awards:

- JavaCup: Category "Internet/Web Agents", Group Winner (US$ 75.000)
- Java Applet Rating Service, Top 1 % Web Applet, Top 10 Web Applet
- Gamelan Featured Applet, Gamelan Cool Applet

[12] Web sites using Promondia to provide chat services include

- http://www.women.com
- http://www.orientation.com
- http://www.worldkids.net
- http://www.surria.com
- http://www.chip.de
- http://www.mainpost.de

summary is available at http://www.cpsc.ucalgary.ca/projects/grouplab/papers/CSCWVideoPapers/GroupWebPaper.ps

[20] P. England, R. Allen and R. Underwood, RAVE: Real-time services for the Web, in: *Proc. of the 5th Int. World Wide Web Conf.* (6–10 May 1996, Paris, France), *Computer Networks and ISDN Systems*, 28(7–11): 1547–1558, 1996, http://www5conf.inria.fr/fich_html/papers/

[21] S. Jacobs, M. Gebhardt, S. Kethers and W. Rzasa, Filling HTML forms simultaneously: CoWeb — architecture and functionality, in: *Proc. of the 5th Int. World Wide Web Conf.* (6–10 May 1996, Paris, France), *Computer Networks and ISDN Systems*, 28(7–11): 1385-1395, 1996, http://www5conf.inria.fr/fich_html/papers/P43/Overview.html

[22] M. Walther, Primitives for group communications in electronic meeting systems, Dissertation, Bond University, Canberra, Australia, 1995.

[23] M. Walther, Supporting development of synchronous collaboration tools on the Web with GroCo, in: *Proc. of the ERCIM Workshop on CSCW and the Web* (St. Augustin, Germany, Feb. 1996), http://orgwis.gmd.de/projects/W4G/proceedings/groco.html

[24] T. Kindberg, G. Coulouris, J. Dollimore and J. Heikkinen, Sharing objects over the Internet: the Mushroom approach, in: *Proc. of Global Internet 96* (London, UK, Nov. 1996).

[25] T. Kindberg, Mushroom: a framework for collaboration and interaction across the Internet, in: *Proc. of the ERCIM Workshop on CSCW and the Web* (St. Augustin, Germany, Feb. 1996), http://orgwis.gmd.de/projects/W4G/proceedings/mushroom.html

[26] P. Ciancarini, A. Knochen, R. Tolksdorf and F. Vitali, PageSpace: An architecture to coordinate distributed applications on the web, in: *Proc. of the 5th Int. World Wide Web Conf.* (6–10 May 1996, Paris, France), *Computer Networks and ISDN Systems*, 28(7–11): 941–952, 1996, http://www5conf.inria.fr/fich_html/papers/P5/Overview.html

Ulrich Gall is a graduate student at the Department of Computer Science at the University of Erlangen-Nürnberg. Since July 1996, he is responsible for 3SOFT's Internet/Java section. His current main research interest are frameworks for distributed, object-oriented systems for applications such as group communication and embedded control.

Franz J. Hauck graduated in computer science at the University of Erlangen-Nürnberg, Germany in 1989. For two years, he was a member of the R&D group of a small company, designing real-time operating systems. He returned to university and got his Ph.D. in 1994. Then, he was a postdoc at the Free University of Amsterdam. Eventually, he went back to the University of Erlangen-Nürnberg in 1996. Currently he is an Assistant Professor. His research interests are Distributed Operating Systems, Object-Oriented Programming, and the World Wide Web.